Laboratoire de Recherche en Technologies de l'Information et de la Communication Genie Electrique (LaTICE)
ENSIT, University of Tunis, Tunisia

A proactive/reactive approach to deal with disturbances in Volunteer Computing Platforms

Adel ESSAFI and ZIED ZAIDI
adel.safi@imag.fr
zied.zaidi@ensit.rnu.tn

July 4, 2016

Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

Internet : a reserve of underexploited resources

Volunteering :

++ Many actors

++ Acceptable performance (technological evolution)

– Latent instability

– Short lived

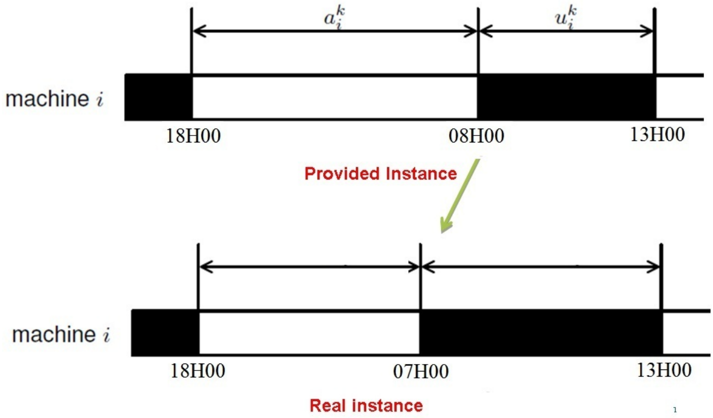
Collaboration :

++ Long-run (Institutional)

++ More or less – stable performance

– Not always evident to place

– Conflict of interest



a_i^k
u_i^k
machine i
18H00
08H00
13H00
Provided Instance
machine i
18H00
07H00
13H00
Real instance

Tasks, machines and objective :

- Tasks : n non preemptive independent tasks of cost p_j
- Machines
  - m machines having speed S_j
  - Machine j is only available during giving intevals
- Objective
  - Makespan (Taken uncertainty into account)
  - Stability : Not altered by disturbances

Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

Uncertainty sources : scheduling parameters

- Scheduling problem entries : Jobs (tasks), Processors (machines), structure (Communicatio, Arrival dates ....)
- Offline schedule : All the parameters are know in advance
- Online schedule: Not all parameters are known a priori
- Scheduling with uncertainties
  - We have prevision on data
  - No complete information

Uncertainty sources

- Destop grids : Uncertainty by nature (volunteer contribution, no control, ...)
- Uncertainty on dates, events,
- Cost (execution time) : depends to the real load
- Structure : Unpredicted tasks, task cancellation, task removal ....
- Communication structure
- Communication delay (correlated to network load)

Dealing with uncertainties

- Robust schedule : Not (or not very much) altered with disturbances

Resolution scheme

STEP 0 : Modelisation
STEP 1 : Static phase : Compute a set of feasible solution
STEP 2 : Dynamic phase : Compute the final (executed) solution

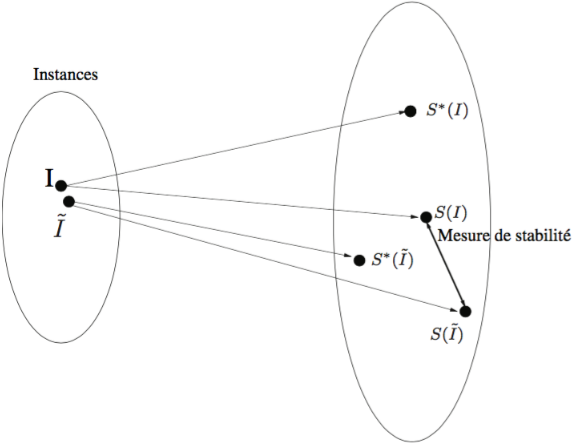
Resolution approach

- **Proactive**
  - Focus on Step 1
  - Design a reference or a set of reference schedules
  - Some **simple** adjustment may be done to keep the schedule feasible
- **Reactive**
  - Focus on Step 2
  - Most decisions are taken at the execution phase
  - Decisions may take long time to be taken
- **Procative/Reactive**
  - Proactive + Sophisticated reaction algorithm (phase 2)
  - Rectification and/or repotimisation

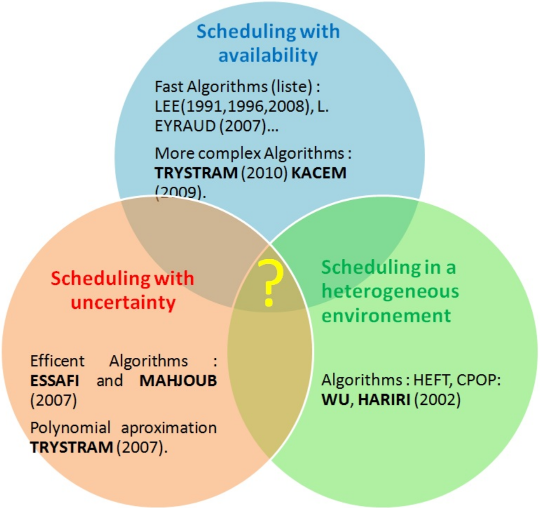
Performance

- Let I be the pretected instance and $\tilde{I}$ the disturbed one
- Let C^*_{max} and $\tilde{C}^*_{mac}$ be the optimal solutions for the predicted and distubed instances
- Let $C_{max}(A)$ and $\tilde{C}_{max}(A)$ the makespan
- Stability : $\rho = max_I \dfrac{\tilde{C}_{max}(A)}{C_{max}(A)}$
- Optimal stability is achieved when $\rho = 1$

Performance



Related work



HEFT principle for DAGs

Phase 1

- Calculation of the priority of each task ($rank_u$), which is based on the average calculation and communication costs.
- The task list is generated by sorting the tasks in decreasing order of $rank_u$.

Phase 2

HEFT Principle

HEFT principle for DAGs

Phase 1

- Calculation of the priority of each task ($rank_u$), which is based on the average calculation and communication costs.
- The task list is generated by sorting the tasks in decreasing order of $rank_u$.

Phase 2

- For most scheduling algorithms, the availability date for a processor p_j is the end of execution of its last task assigned.
- Insertion policy
- Possibility of insertion of a task in an interval of inactivity

Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

HEFT adaptation to availability constraint

- $EST(i,j) = avail(j,i)$: First date in which j can execute task i
- $EFT(i,j) = EST(i,j) + \frac{P_i}{S_j}$:End date of i task if executed on processor j
- $rank_u(i)$: Priority of the task i

HEFT-AC Algorithm

1. Compute the $Rank_u$ and the average cost of processing for all tasks
2. Sort all tasks in order of decreasing values of $Rank_u$.
3. **While** there are unscheduling tasks in the list **do**
5. Select the first task, n_i, from the list for scheduling
6. **For** each processor p_k in the processor-set ($p_k \in Q$) **do**
7. Compute the availablity date for n_i ($avail[p_k, n_i]$)
7. Compute EFT (n_i, p_k) value
8. Assign task n_i to the processor p_k that minimize EFT of task n_i.
9. **endwhile**

What's wrong with HEFT-AC ?

Risks

- Can fill an interval availability entirely
- Uncertainty unawareness ?

Improvement

- We must improve the allocation of tasks to machines using the availability model.

Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

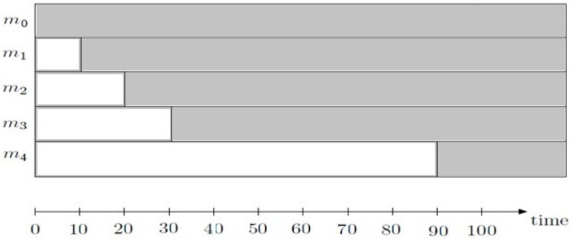
Performance modeling , Kondo et al.

- Identifying correlation between resources based on their availability
- Standard classification algorithm (K-means)
- They were able to identify 5 classes of machines

The average availability of a machine in a grid is a good criterion for the classification

Alpha

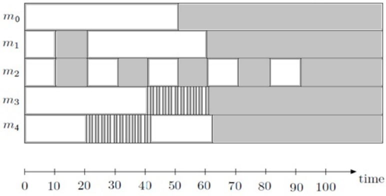
$$alpha[j] = \frac{\sum_{k=1}^{nb_{intervalles[j]}} a_j^k * vitesse[j]}{nb_{intervalles[j]}}$$



Is the average amount of computation in one interval

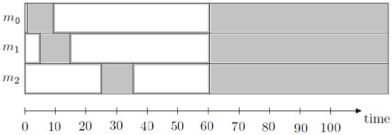
alpha[0]= 0
alpha[1]= 10
alpha[2]= 20
alpha[3]= 30
alpha[4]=90

Drawback whith Alpha



alpha[0]= 50
alpha[1]= 25
alpha[2]= 10
alpha[3]= 50/11 = 4.54
alpha[4]= 50/12 = 4.16

The alpha parameter does not take into account the dispersion pattern of intervals availability.



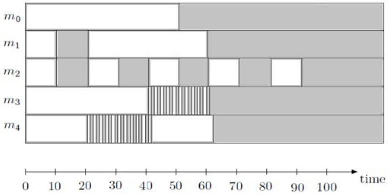
alpha[0]= 25
alpha[1]= 25
alpha[2]= 25

The alpha parameter is insufficient to characterize machines

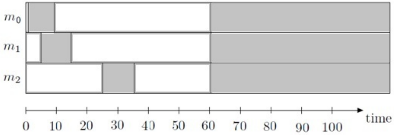
1

Beta

beta $[j]=\sum_{k=0}^{nb\_intervalles[j]}\left(a_j^k-\bar{l}[j]\right)^2$ **Is the variance between availability intervals of machine j**



Valeur de alpha	Valeur de beta
alpha[0]= 50	beta[0]=0
alpha[1]= 25	beta[1]=450
alpha[2]= 10	beta[2]=0
alpha[3]= 50/11 = 4.54	beta[3]=138.27
alpha[4]= 50/12 = 4.16	beta[4]=54.69



Valeur de alpha	Valeur de beta
alpha[0]= 25	beta[0]=1152
alpha[1]= 25	beta[1]=1058
alpha[2]=25	beta[2]=0

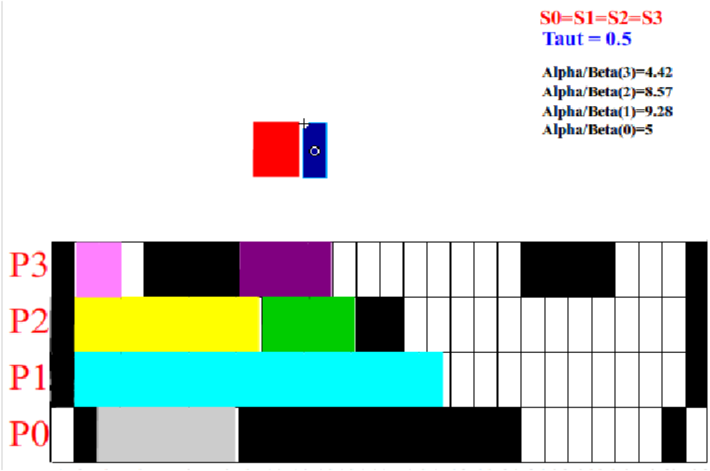
1

HEFT-ACU Algorithm

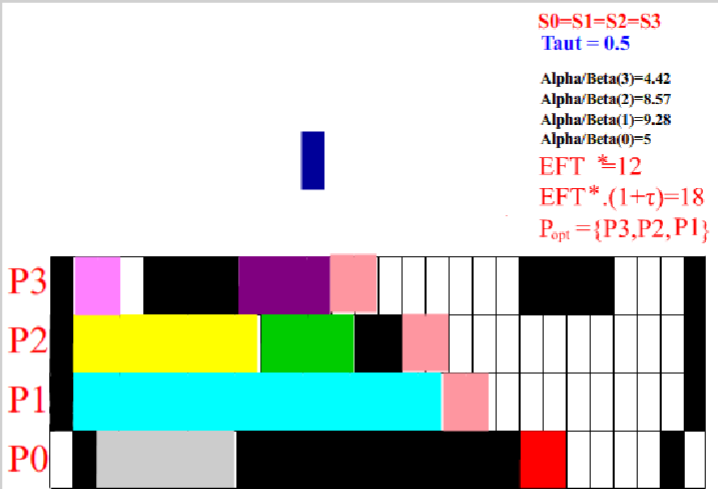
1. Compute the $Rank_u$ for all tasks
2. Compute alpha[j] and beta[j] for all processors.
3. Sort all tasks in order of decreasing values of $Rank_u$. (Longest First)
6. **While** there are unscheduled tasks in the list **do**
7. Select the first task, ni from the list
8. Compute $EFT^*(n_i) = Min_{j \in 1..m}(EFT\ (n_i, p_j)$
9. Let P_{cand} the list of processor p_j that $EFT\ (n_i, p_j) \leq EFT^*\ (n_i,)\ *(1+\tau)$
10. Assign task n_i to processor p_j from P_{cand} such as alpha[j] /beta[j] is the maximum
11. **endwhile**

Online execution : Adjustments to keep the schedule feasible

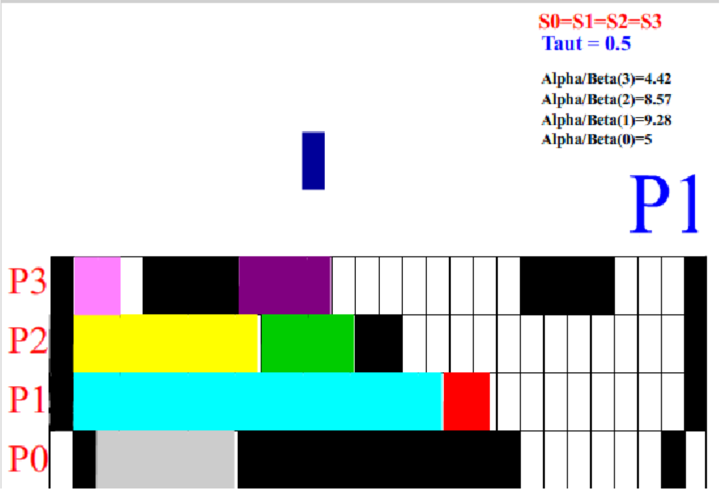
Exemple for Tasks allocation in HEFT-ACU



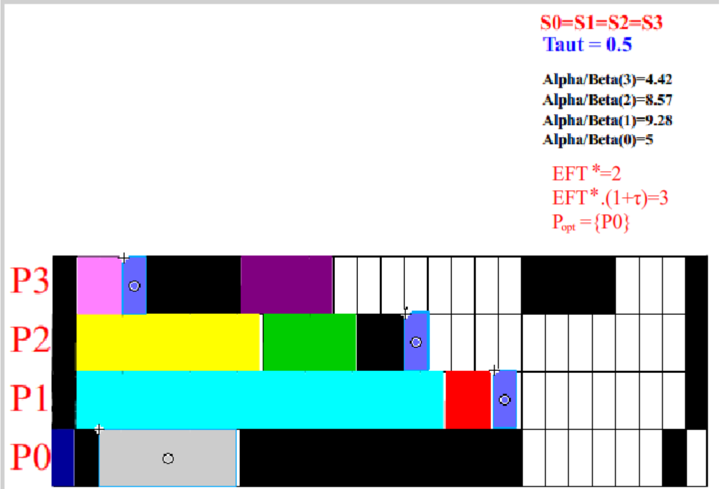
Exemple for Tasks allocation in HEFT-ACU



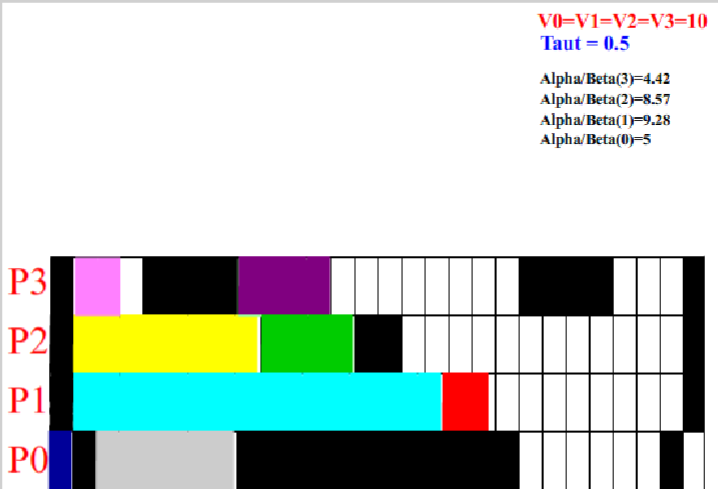
Exemple for Tasks allocation in HEFT-ACU



Exemple for Tasks allocation in HEFT-ACU



Exemple for Tasks allocation in HEFT-ACU



Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

Limits of the proactive approch

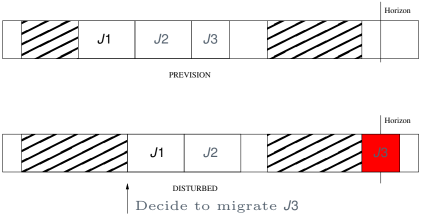
- Application : a set of tasks
- execution : all tasks must be executed
- No garentee : in disturbed environments
- Solution : Duplucate and/or migrate
- Our approach : Duplicate tasks if their completion time exceed a limit

Model

- Desktop grid
- No control on the host when unavailable
- No preemtion
- No migration cost (since the task will be sent from the broker to the worker)

Model

- For all machine j, Let $C^j = maxC_i$ for jobs i scheduled on machine j
- Define $horizon^j = \frac{3}{2}C^j$
- Purpose : In the disturbed execution, all tasks must be scheduled before $Horizon^j$



Target Processor

- At the begin of the current availability : identify jobs that exceeds $Horizon^j$
- Re-schedule these tasks to a new processor
- New processor is chosen in order to complete the tasks before *the horizon*
- Set of CL candidate processors
- Selection Criterion : Processor with highest stability (i.e $\frac{alpha}{beta}$)

Rescheduling tasks

- Phase 1 : Offline schedule (HEFT-AC)
- Phase 2 : Online execution with limited reaction mecanism base on delaying start execution time

Rescheduling tasks

- Phase 1 : Offline schedule (HEFT-AC)
- Phase 2 : Online execution with limited reaction mecanism base on delaying start execution time

Rescheduling tasks

- Phase 1 : Offline schedule (HEFT-AC)
- Phase 2 : Online Execution with sophisticated reaction mecanism
- Phase 3 : A decision of migrating the tasks is taken online
- Phase 3 : minimize the lateness of the completion time

Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

Execution Scenario

Scenario 1: (without disturbance) Tasks are executed using the planned dates.

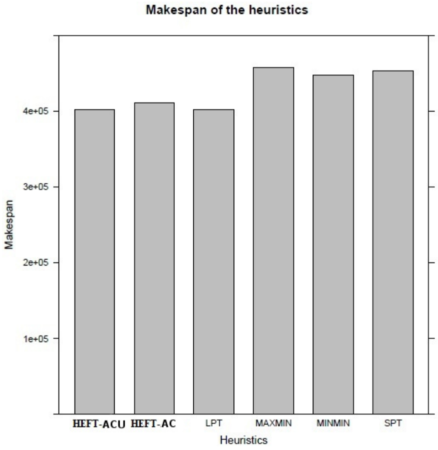
Scenario 2: (with disturbance) Potential interruptions of tasks are handled by a local re scheduling mechanism on the same processor.

We designed a specific simulator which supports machine profiles, task profile and disturbance in availability intervals

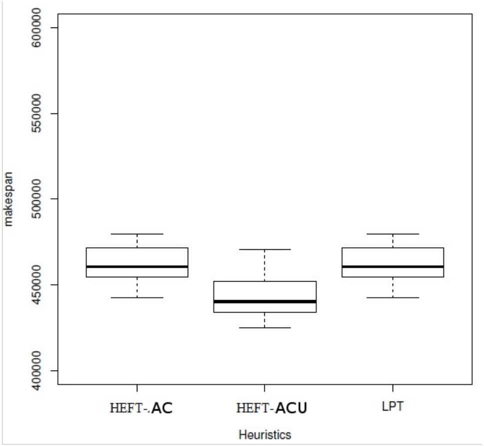
Study of performance of heuristics

- Six algorithms are studied :
  - LPT
  - SPT
  - MinMin
  - MaxMin
  - HEFT-AC
  - HEFT-ACU
- Test are performed on 10 different instances
  - 10000 tasks
  - 1000 machines
  - τ = 0.2 (empiricaly chosen)
  - Each instance is disturbed 30 times

Performance Comparison between Heuristics without disturbance



Performance Comparison between Heuristics with disturbance



Outline

Introduction

State of the art

Adaptation of HEFT to the availability constraint

Stable algorithm for disturbed environments

More reactive approach

Validation and Empirical study

Conclusion

Conclusion

- We are considering uncertain Desktop grid platforms
- We adapted HEFT to schedule tasks within the schedule instead of only at the end
- We use the average and variance of the length of availability intervals to characterise the most stable machines
- HEFT-ACU is the most stable evaluated algorithm

Conclusion

- Limied reaction mecanism leads to non-executed jobs
- We moved from *proactive* approach, to a proactive/reactive approach
- Primary results : Migration improve distrubed makespan in disturbed envirnments with at least 5%

Some references ....

Billaut J.C., Moukrim A. and Sanlaville E.
Flexibilité et robustesse en ordonnancement.
Hermès, 2005.

Essafi A. , Trystram D. and Zaidi Z.
An Efficient Algorithm for Scheduling Jobs in Volunteer Computing Platforms
HCW - Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, 68–76,2014.

The End

Thank you for your attention